Computing Kitahara Mizuno's bound on the number of basic feasible solutions generated with the simplex algorithm

doi: 10.1007/s11590-018-1276-4

Computing Kitahara-Mizuno's bound on the number of basic feasible solutions generated with the simplex algorithm

Takahito Kuno*, Yoshio Sano†, and Takahiro Tsuruda‡

Graduate School of Systems and Information Engineering

University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

September, 2017
Revised: February, 2018

Abstract

This paper addresses an upper bound derived by Kitahara and Mizuno [10] on the number of basic feasible solutions of a linear program generated with the simplex algorithm. Their bound includes two parameters δ and γ, which are respectively the minimum and the maximum values of positive components in all basic feasible solutions. We show that δ is NP-hard to determine while γ can be computed in polynomial time. We also report some numerical results using alternative parameters for δ and γ.

Key words: Linear programming, simplex algorithm, number of iterations, basic feasible solutions, NP-hardness.

1 Introduction

While over thirty years have passed since the first two polynomial-time algorithms were published [7,9], it is still an open question whether linear programming generally admits a strongly polynomial-time algorithm whose number of iterations depends only on the number of variables n and the number of constraints m. At present, the simplex algorithm is conjectured to serve as a strongly polynomial-time algorithm, by devising an ingenious pivot selection rule [1]. As if to support this, Ye proved in 2011 [16] that the simplex algorithm solves a linear program derived from the Markov decision problem with a fixed discount rate in strongly polynomial time, even under the usual Dantzig's pivot selection rule. The simplex algorithm applied to the Markov decision problem is also dealt with in [5, 12]. In 2013, Kitahara and Mizuno [10] extended Ye's result to general linear programs and showed that the number of

*Partially supported by a Grant-in-Aid for Scientific Research (C) 16K00028 from the Japan Society for the Promotion of Sciences. E-mail: takahito@cs.tsukuba.ac.jp

†Partially supported by a Grant-in-Aid for Young Scientists (B) 15K20885 from the Japan Society for the Promotion of Sciences. E-mail: sano@cs.tsukuba.ac.jp

‡Current affiliation: Cybozu, Inc.

different basic feasible solutions generated before the simplex algorithm terminates is bounded from above by $n\lceil(m\gamma/\delta)\log(m\gamma/\delta)\rceil$, where δ and γ are respectively the minimum and the maximum values of positive components in all basic feasible solutions of the problem. In the case of the Markov decision problem with discount rate θ, we have $\delta \geq 1$ and $\gamma \leq m/(1-\theta)$, and hence Kitahara-Mizuno's upper bound is strongly polynomial in m and n. So, what kind of size is this upper bound for general linear programs? The purpose of this paper is to clarify this simple question.

In Section 2, after outlining the behavior of the simplex algorithm, we introduce Kitahara-Mizuno's analysis on the simplex algorithm in some detail. In Section 3, to make actual measurements of δ and γ, we discuss how to obtain these parameters. We then show that δ is NP-hard to determine while γ can be computed in polynomial time. In Section 4, we introduce easily computable substitutes for δ and γ, and report the numerical results using those for 20 instances selected from the Netlib collection [14].

2 Theoretical behavior of the simplex algorithm

The problem considered in this paper is a linear program of the standard form:

$$\left| \begin{array}{ll} \text{minimize} & \mathbf{c}\,\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}, \end{array} \right. \tag{1}$$

where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_n] \in \mathbb{R}^{m\times n}$ is full row-rank, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c}^\mathsf{T} \in \mathbb{R}^n$. For simplicity, let us assume that the feasible set is nonempty and bounded. Then (1) has an optimal basic solution $\mathbf{x}^*$ with value $z^* = \mathbf{c}\mathbf{x}^*$. The dual problem of (1) is written as

$$\left| \begin{array}{ll} \text{maximize} & \mathbf{b}^\mathsf{T}\mathbf{y} \\ \text{subject to} & \mathbf{A}^\mathsf{T}\mathbf{y} + \mathbf{s} = \mathbf{c}^\mathsf{T}, \quad \mathbf{s} \geq \mathbf{0}, \end{array} \right. \tag{2}$$

where $\mathbf{s} \in \mathbb{R}^n$ is a vector of slack variables. By the duality theorem (see e.g., [2, 4]), this problem also has an optimal solution $(\mathbf{y}^*, \mathbf{s}^*)$ with value $\mathbf{b}^\mathsf{T}\mathbf{y}^* = z^*$.

Overview of the simplex algorithm

Let $\mathbf{B} \in \mathbb{R}^{m\times m}$ be a submatrix of $\mathbf{A}$ such that $\mathbf{B}$ is invertible and $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, and let $\mathbf{N} \in \mathbb{R}^{m\times(n-m)}$ denote the remaining part of $\mathbf{A}$. Also let B and N denote their column index sets. After swapping some columns of the matrix and vectors as follows

$$\mathbf{A} = [\mathbf{B}, \mathbf{N}], \quad \mathbf{c} = [\mathbf{c}_B, \mathbf{c}_N], \quad \mathbf{x}^\mathsf{T} = [\mathbf{x}_B^\mathsf{T}, \mathbf{x}_N^\mathsf{T}],$$

we have a linear system, called a *dictionary* of (1):

$$\left| \begin{array}{l} \mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N \\ z = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} - \bar{\mathbf{c}}_N\mathbf{x}_N, \end{array} \right. \tag{3}$$

where $\bar{\mathbf{c}}_N = \mathbf{c}_N - \mathbf{c}_B\mathbf{B}^{-1}\mathbf{N}$. Usually, $\mathbf{x}_B$ and $\mathbf{x}_N$ are referred to as the vectors of *basic* and *nonbasic variables*, respectively. Since $\mathbf{B}^{-1}\mathbf{b} \geq \mathbf{0}$, the pair $(\mathbf{x}_B, \mathbf{x}_N) = (\mathbf{B}^{-1}\mathbf{b}, \mathbf{0})$ gives a basic feasible solution to (1). If $\bar{\mathbf{c}}_N \leq \mathbf{0}$, the pair is an optimal solution to (1). Otherwise, there exists a positive component, say $\bar{c}_s$, in $\bar{\mathbf{c}}_N$. If the value of variable x_s increases from zero to $v \geq 0$, the objective function value z decreases accordingly by $\bar{c}_s v$. Once some row, say x_r, of $\mathbf{x}_B = \mathbf{B}^{-1} - \mathbf{B}^{-1}\mathbf{a}_s v$ reaches zero, the columns $\mathbf{a}_r$ of $\mathbf{B}$ and $\mathbf{a}_s$ of $\mathbf{N}$ are swapped and the dictionary (3) is updated. The simplex algorithm repeats these steps and generates a sequence of basic feasible solutions $\mathbf{x}^1, \ldots, \mathbf{x}^k, \ldots$ of (1), starting from an initial basic feasible solution $\mathbf{x}^0$, until $\bar{\mathbf{c}}_N \leq \mathbf{0}$ holds.

The step of swapping $\mathbf{a}_r$ and $\mathbf{a}_s$ is referred to as *pivoting*. If there are multiple positive components in $\bar{\mathbf{c}}_N$, we need to select an appropriate one from among them to perform pivoting. Many selection rules have been proposed so far (see [15]), and among others, the *largest coefficient rule*, also known as Dantzig's rule, and the *largest improvement rule* are the two most popular selection rules. The former suggested by Dantzig [4] simply chooses as $\bar{c}_s$ the largest component in $\bar{\mathbf{c}}_N$. The latter first calculates

$$v_j = \min\{b_i/a_{ij} \mid a_{ij} > 0,\ i \in B\}$$

for each $j \in N$ such that $\bar{c}_j > 0$; and then it adopts $\bar{c}_j$ with the largest $\bar{c}_j v_j$ as $\bar{c}_s$. Therefore, while it requires more computational effort, the largest improvement rule can reduce the objective function value z more than the largest coefficient rule does for each pivoting step. Hereafter, the simplex algorithm is assumed to follow either the largest coefficient or the largest improvement rule.

THE ANALYSIS OF KITAHARA-MIZUNO [10]

Let δ and γ respectively denote the minimum and the maximum values of positive components in all basic feasible solutions of (1). Namely, each basic feasible solution $\mathbf{x}^k$ encountered in the simplex algorithm satisfies

$$\delta \leq x_j^k \leq \gamma \quad \text{if } x_j^k \neq 0. \tag{4}$$

Suppose that (3) is the dictionary corresponding to $\mathbf{x}^k$, and that $\mathbf{x}^k \neq \mathbf{x}^*$. Naturally, $\mathbf{x}^*$ and z^* still satisfy the system (3). Hence, by noting that $\mathbf{x}^*$ has at most m positive components, we have

$$z^* = \mathbf{c}_B\mathbf{B}^{-1}\mathbf{b} - \bar{\mathbf{c}}_N\mathbf{x}_N^* \geq \mathbf{c}\mathbf{x}^k - \bar{c}_s\mathbf{e}\mathbf{x}_N^* \geq \mathbf{c}\mathbf{x}^k - \bar{c}_s m\gamma \tag{5}$$

if $\bar{c}_s$ is chosen by the largest coefficient rule, where $\mathbf{e} \in \mathbb{R}^{n-m}$ is the all-ones vector. The dictionary (3) is updated by increasing the value of variable x_s to x_s^{k+1} while keeping other nonbasic variables equal to zero. This, together with (5), implies

$$\mathbf{c}\mathbf{x}^k - \mathbf{c}\mathbf{x}^{k+1} = \bar{c}_s x_s^{k+1} \geq \bar{c}_s\delta \geq \frac{\delta}{m\gamma}(\mathbf{c}\mathbf{x}^k - z^*)$$

as long as $x_s^{k+1} \neq 0$. Note that the right-hand-side of the last inequality is also a lower bound on the decrement of the objective function value when the largest improvement rule is adopted. Thus we have the first key result (Theorem 1 in [10]):

Lemma 2.1 *If* $\mathbf{x}^k \neq \mathbf{x}^{k+1}$, *then*

$$\mathbf{c}\mathbf{x}^{k+1} - z^* \leq \left(1 - \frac{\delta}{m\gamma}\right)(\mathbf{c}\mathbf{x}^k - z^*).$$

From the feasibility of $\mathbf{x}^k$ and $(\mathbf{y}^*, \mathbf{s}^*)$, we have

$$\mathbf{c}\mathbf{x}^k - z^* = \mathbf{c}\mathbf{x}^k - \mathbf{b}^\mathsf{T}\mathbf{y}^* = (\mathbf{A}^\mathsf{T}\mathbf{y}^* + \mathbf{s}^*)^\mathsf{T}\mathbf{x}^k - (\mathbf{A}\mathbf{x}^k)^\mathsf{T}\mathbf{y}^* = \sum_{j \in B} s_j^* x_j^k, \tag{6}$$

by noting that $x_j^k = 0$ for each $j \in N$. Since $s_j^* x_j^k \geq 0$ for each $j \in B$, there exists an index $t \in B$ such that $s_t^* x_t^k \geq (\mathbf{c}\mathbf{x}^k - z^*)/m$. In a way similar to (6), we have $\mathbf{c}\mathbf{x}^i - z^* \geq s_t^* x_t^i$ for each i. These imply the second key result (Lemma 2 in [10]):

Lemma 2.2 *If* $\mathbf{x}^k \neq \mathbf{x}^*$, *then there exists an index* $t \in B$ *such that*

$$x_t^k > 0, \quad s_t^* \geq \frac{1}{m x_t^k}(\mathbf{c}\mathbf{x}^k - z^*).$$

Moreover, for each i, the basic feasible solution $\mathbf{x}^i$ *satisfies*

$$x_t^i \leq m \frac{\mathbf{c}\mathbf{x}^i - z^*}{\mathbf{c}\mathbf{x}^k - z^*} x_t^k.$$

Let $\bar{k}$ denote the number of different basic feasible solutions in $\{\mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^k\}$. We see from Lemma 2.2 that there exists a basic variable x_t in $\mathbf{x}^0$ such that $x_t^0 > 0$ and

$$x_t^k \leq m \frac{\mathbf{c}\mathbf{x}^k - z^*}{\mathbf{c}\mathbf{x}^0 - z^*} x_t^0.$$

We also see from Lemma 2.1 that

$$\mathbf{c}\mathbf{x}^k - z^* \leq \left(1 - \frac{\delta}{m\gamma}\right)^{\bar{k}} (\mathbf{c}\mathbf{x}^0 - z^*).$$

These inequalities, together with (4), imply

$$x_t^k \leq m \left(1 - \frac{\delta}{m\gamma}\right)^{\bar{k}} x_t^0 \leq m\gamma \left(1 - \frac{\delta}{m\gamma}\right)^{\bar{k}}.$$

The right-hand-side of the second inequality is less than δ if

$$\bar{k} > m \frac{\gamma}{\delta} \log\left(m \frac{\gamma}{\delta}\right),$$

because $\log(1-x) \leq -x$ holds in general if $x < 1$. For such a $\bar{k}$, we have $x_t^k < \delta$, i.e., $x_t^k = 0$, and the variable x_t remains zero after the kth iteration. As in the case of $\mathbf{x}^0$, if $\mathbf{x}^k \neq \mathbf{x}^*$, there exists a positive-valued variable in $\mathbf{x}^k$, which vanishes if the simplex algorithm generates $\bar{k}$ different basic feasible solutions at most. Since the number of variables is n, we have the last key result (Theorem 2 [10]):

Theorem 2.3 *The simplex algorithm finds an optimal solution* $\mathbf{x}^*$ *of the problem (1) after generating* $n\lceil(m\gamma/\delta)\log(m\gamma/\delta)\rceil$ *different basic feasible solutions at most.*

Kitahara et al. [11] has performed a similar analysis for the upper bounding simplex algorithm (see e.g., [2, 4]).

3 How to derive the ratio of δ to γ for general linear programs

Let us denote the upper bound given in Theorem 2.3 as

$$U(m,n,\delta,\gamma) = n\left\lceil m\frac{\gamma}{\delta}\log\left(m\frac{\gamma}{\delta}\right)\right\rceil.$$

As is obvious from this bound, the simplex algorithm serves as a strongly polynomial-time algorithm for a linear program if the problem is nondegenerate and the ratio of γ to δ is bounded from above by a polynomial in m and n. A typical case is where all basic feasible solutions are integral, and realized when $\mathbf{A}$ is totally unimodular and $\mathbf{b}$ is integral. Then, for general linear programs, how large is $U(m,n,\delta,\gamma)$? To answer the question, in this section, we discuss methods of computing the values γ and δ without assuming any special structures on the problem (1).

Computation of the value γ

With respect to γ, it is not hard to compute the value. In fact, γ can be determined after solving the following linear program for $j = 1,\ldots,n$:

$$\left|\begin{array}{ll} \text{maximize} & x_j \\ \text{subject to} & \mathbf{Ax} = \mathbf{b},\quad \mathbf{x} \geq \mathbf{0}. \end{array}\right. \tag{7}$$

The feasible set of (1) has been assumed to be nonempty and bounded, and hence (7) with the same feasible set has an optimal solution, say $\mathbf{x}^j$. The value of γ is given by the maximum of x_j^j for $j = 1,\ldots,n$. Since polynomial-time algorithms are available for (1), and hence for (7) [7, 9], the total complexity of computing γ is also bounded by a polynomial in the input length of (7), or (1).

Computation of the value δ

Unfortunately, in contrast to γ, computing the value δ is intractable. In fact, the following well-known NP-complete problem (see also [6]) reduces in polynomial time to the problem of

finding a basic feasible solution with a component equal to δ.

PARTITION [8]
INSTANCE: n positive integers a_j, $j = 1, \ldots, n$.
QUESTION: Is there a subset $S \subset \{1, \ldots, n\}$ such that $\sum_{j \in S} a_j = \sum_{j \notin S} a_j$?

Before proceeding the discussion, let us define our problem more precisely:

FINDING δ
INPUT: An integer $m \times n$ matrix $\mathbf{A}$ and an integer m-vector $\mathbf{b}$.
OUTPUT: If any, a basic feasible solution with the minimum positive component in all basic feasible solutions of the system:

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}.$$

Theorem 3.1 *The problem FINDING δ is NP-hard.*

Proof. Let us consider an arbitrary instance of the problem PARTITION. Without loss of generality, we can assume $a_j > 1$ for some j. Otherwise, the problem is trivial to solve: the answer is "yes" if and only if n is even. We can also assume that $\sum_{j=1}^{n} a_j$ is an even integer, because otherwise the answer is "no". Let $b = \sum_{j=1}^{n} a_j/2$ and define a linear system:

$$\left| \begin{array}{l} \displaystyle\sum_{j=1}^{n} a_j x_j = b \\ x_j + x_{n+j} = 1, \quad j = 1, \ldots, n \\ \mathbf{x} \geq \mathbf{0}, \end{array} \right. \tag{8}$$

where $\mathbf{x} = (x_1, \ldots, x_{2n})^{\mathsf{T}}$. Note that each basic feasible solution to this system is obtained by fixing either x_j or x_{n+j} to zero for all j except one. Therefore, the variables of at most one pair (x_j, x_{n+j}) can be fractions in each basic feasible solution. If the system (8) has a basic feasible solution $\mathbf{x}'$ without such a fraction variable pair, we can answer "yes" along with the certification $S = B \cap \{1, \ldots, n\}$, where B is the index set of basic variables in $\mathbf{x}'$. In order to check it, we solve a slightly perturbed system:

$$\left| \begin{array}{l} \alpha \displaystyle\sum_{j=1}^{n} a_j x_j = \alpha b + 1 \\ x_j + x_{n+j} = 1, \quad j = 1, \ldots, n \\ \mathbf{x} \geq \mathbf{0}, \end{array} \right. \tag{9}$$

where α is any integer larger than $\max\{a_j \mid j = 1, \ldots, n\} + 1$. First, we will show that there exists a surjection from the set of basic feasible solutions of (9) to the set of basic feasible solutions of (8).

Let $\mathbf{x}'$ be any basic feasible solution of (8), and let $r \in \arg\min\{x'_j \mid j \in B \cap \{1, \ldots, n\}\}$. There are three cases to consider.

Case 1: $0 < x'_r < 1$. Since a_j's and b are all integers, we have

$$\frac{1}{a_r} \leq x'_r \leq 1 - \frac{1}{a_r}, \quad \frac{1}{a_r} \leq x'_{n+r} \leq 1 - \frac{1}{a_r}. \tag{10}$$

Let

$$x''_r = x'_r + \frac{1}{\alpha a_r}, \quad x''_{n+r} = x'_{n+r} - \frac{1}{\alpha a_r}, \quad x''_j = x'_j \text{ for each } j \notin \{r,\, n+r\}. \tag{11}$$

Then $\mathbf{x}''_r + \mathbf{x}''_{n+r} = 1$, and we have $0 \leq x''_r, x''_{n+r} \leq 1$ since $0 < 1/(\alpha a_r) < 1/a_r$. We see that $\mathbf{x}''$ is a basic feasible solution of (9).

Case 2: $x'_r = 0$. Let

$$x''_r = \frac{1}{\alpha a_r}, \quad x''_{n+r} = 1 - \frac{1}{\alpha a_r}, \quad x''_j = x'_j \text{ for each } j \notin \{r,\, n+r\}. \tag{12}$$

Again, $\mathbf{x}''$ is a basic feasible solution of (9).

Case 3: $x'_r = 1$. There exists an index $s \in \{1, \ldots, n\}$ such that $x'_s = 0$ and $x'_{n+s} = 1$; otherwise, $\sum_{j=1}^{n} a_j x'_j = \sum_{j=1}^{n} a_j = b = \sum_{j=1}^{n} a_j/2$, which contradicts the fact that a_j's are positive integers. Let

$$x''_s = \frac{1}{\alpha a_s}, \quad x''_{n+s} = 1 - \frac{1}{\alpha a_s}, \quad x''_j = x'_j \text{ for each } j \notin \{s,\, n+s\}. \tag{13}$$

Then $\mathbf{x}''$ is a basic feasible solution of (9), as in case 2.

Thus, in any case, there exists a basic feasible solution of (9) corresponding to $\mathbf{x}'$.

Now, let $\mathbf{x}''$ be any basic feasible solution to (9). Let $r \in \arg\min\{x''_j \mid j \in B \cap \{1, \ldots, n\}\}$. Then $0 < x''_r < 1$, and hence we have

$$\frac{1}{\alpha a_r} \leq x''_r \leq 1 - \frac{1}{\alpha a_r}, \quad \frac{1}{\alpha a_r} \leq x''_{n+r} \leq 1 - \frac{1}{\alpha a_r}.$$

Let

$$x'_r = x''_r - \frac{1}{\alpha a_r}, \quad x'_{n+r} = x''_{n+r} + \frac{1}{\alpha a_r}, \quad x'_j = x''_j \text{ for each } j \notin \{r,\, n+r\}.$$

Then $\mathbf{x}'$ is a basic feasible solution of (8).

The above discussion implies that each basic feasible solution $\mathbf{x}'$ of (8) in Case 1 corresponds to some basic feasible solution $\mathbf{x}''$ of (9) in a one-to-one fashion. From (10) and (11), we have

$$x''_r \geq \frac{1}{a_r} + \frac{1}{\alpha a_r} > \frac{1}{\alpha}, \quad x''_{n+r} \geq \frac{1}{a_r} - \frac{1}{\alpha a_r} > \frac{1}{\alpha},$$

by noting that $\alpha > \max\{a_j \mid j = 1, \ldots, n\} + 1$. As a result, all positive components in $\mathbf{x}''$ are greater than $1/\alpha$. In contrast to this, if $\mathbf{x}''$ corresponds to a basic feasible solution of (8) in Case 2 or 3, there exists a fractional component x''_r, or x''_s, such that

$$x''_r = \frac{1}{\alpha a_r} \leq \frac{1}{\alpha}, \quad x''_s = \frac{1}{\alpha a_s} \leq \frac{1}{\alpha}.$$

Therefore, the answer to PARTITION is "yes" if and only if the output of FINDING δ for the system (9) is less than $1/\alpha$. □

4 Numerical results

As suggested by Theorem 3.1, the only surest way to obtain the exact value of δ is currently enumeration of all basic feasible solutions. However, an upper bound on δ is easy to obtain with some additional effort in the course of solving (1) by the simplex algorithm. The analysis of Kitahara-Mizuno [10] reviewed in Section 2 was conducted on the basis of the relationship (4), i.e., $\delta \leq x_j^k \leq \gamma$ if $x_j^k \neq 0$, satisfied by each basic feasible solution $\mathbf{x}^k$ encountered in the simplex algorithm. Suppose the algorithm terminates after generating K basic feasible solutions, and let

$$\delta^+ = \min\{x_j^k \mid x_j^k > 0,\ j = 1,\ldots,n;\ k = 0,1,\ldots,K\}$$
$$\gamma^- = \max\{x_j^k \mid x_j^k > 0,\ j = 1,\ldots,n;\ k = 0,1,\ldots,K\}.$$

Then, naturally, as in (4), each basic feasible solution $\mathbf{x}^k$ satisfies

$$\delta^+ \leq x_j^k \leq \gamma^- \quad \text{if } x_j^k \neq 0.$$

Even replacing δ and γ by δ^+ and γ^-, respectively, we can develop the same argument as in Section 2, and derive an upper bound:

$$U(m,n,\delta^+,\gamma^-) = n\left\lceil m\frac{\gamma^-}{\delta^+}\log\left(m\frac{\gamma^-}{\delta^+}\right)\right\rceil \tag{14}$$

on the number of different basic feasible solutions generated with the simplex algorithm. Since we need to generate K basic feasible solutions beforehand, this result is somewhat preposterous. Nevertheless, due to limited computational resources, we calculated this bound, instead of the bound $U(m,n,\delta,\gamma)$ of Kitahara-Mizuno, and compared it with the actual number of different basic feasible solutions, denoted by $\overline{K}$, generated with the simplex algorithm. For this purpose, we modified an existing computer code of the simplex algorithm provided in [3], adopting the largest coefficient (Dantzig's) rule. We then selected from [14] 20 instances of sizes between $(m,n) = (27,59)$ and $(990,2361)$, without considering their problem structures, and solved them on an Intel Core i7-4790 (3.60GHz).

Our computational results are summarized in Table 1, which lists the problem size ($m \times n$), the number of total iterations (*# iterations*), the number of generated different basic feasible solutions ($\overline{K}$), the minimum positive component (δ^+), the maximum positive component (γ^-) in the K basic feasible solutions, and the upper bound defined in (14) ($U(m,n,\delta^+,\gamma^-)$) when the simplex algorithm solved each instance (*name*). For a reference, the last column gives the value:

$$V(m,n) = \binom{n-\lceil (n-m+1)/2\rceil}{m} + \binom{n-\lceil (n-m+2)/2\rceil}{m},$$

which is known to be an upper bound on the number of vertices in the feasible set of (1) [13]. We see from Table 1 that $U(m,n,\delta^+,\gamma^-)$ is a considerable overestimate of $\overline{K}$, and even of the number of total iterations, while it is far less than $V(m,n)$ for each instance. Since $\delta \leq \delta^+$ and $\gamma \geq \gamma^-$, the original $U(m,n,\delta,\gamma)$ would be a yet larger overestimate. Although $U(m,n,\delta,\gamma)$ is a powerful guarantee that the simplex algorithm serves as a polynomial-time algorithm for some special structured problems, it might be too weak to estimate the number of iterations needed to solve general problems with no favorable structure. Anyway, to unravel the relationship between problem structures and $U(m,n,\delta,\gamma)$, we need further theoretical and empirical research.

References

[1] Adler, I., C. Papadimitriou, and A. Rubinstein, "On simplex pivoting rules and complexity theory", *Proceedings of the 17th International Conference on Integer Programming and Combinatorial Optimization* (2014), 13–24.

[2] Chvátal, V., *Linear Programming*, WH Freeman (NY, 1983).

[3] COIN-OR, *Linear Programming Solver* (https://projects.coin-or.org/Clp, 2012).

[4] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press (NJ, 1963).

[5] Fearnley, J., and R. Savani, "The complexity of the simplex method", *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing (STOC'15)* (2015), 201–208.

[6] Garey, M.R., and D.V. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, WH Freeman (NY, 1979).

[7] Karmarkar, N., "A new polynomial algorithm for linear programming", *Combinatorica* **4** (1984), 373–395.

[8] Karp, R.M., "Reducibility among combinatorial problems", in R.E. Miller, and J.W. Thatcher (eds.), *Complexity of Computations*, Plenum Press (NY, 1972), 85–103.

[9] Khachiyan, L.G., "Polynomial algorithms in linear programming", *USSR Computational Mathematics and Mathematical Physics* **20** (1980), 53–72.

[10] Kitahara, T., and S. Mizuno, "A bound for the number of different basic solutions generated by the simplex method", *Mathematical Programming A* **137** (2013), 579–586.

[11] Kitahara, T., S. Mizuno, and T. Matsui, "On the number of solutions generated by Dantzig's simplex method for LP with bounded variables", *Pacific Journal of Optimization* **8** (2012), 447–455.

[12] Littman, M.L., T.L. Dean, and L.P. Kaelbling, "On the complexity of solving Markov decision problems", *Proceedings of the eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI'95)* (1995), 394–402.

[13] McMullen, P., "The maximum numbers of faces of a convex polytope", *Mathematika* **17** (1970), 179–184.

[14] Netlib, *Netlib Collection of LP Problems in MPS Format* (http://www.netlib.org/lp/data, 1988).

[15] Terlaky, T., and S. Zhang, "Pivot rules for linear programming: a survey on recent theoretical developments", *Annals of Operations Research* **(**46) (1993), 203–233.

[16] Ye, Y., "The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate", *Mathematics of Operations Research* **36** (2011), 593–603.

Table 1: Numerical results on the simplex algorithm with the largest coefficient rule

name	$m \times n$	*# iterations*	$\overline{K}$	δ^+	γ^-	$U(m,n,\delta^+,\gamma^-)$	$V(m,n)$
afiro	27×59	16	7	2.615	440.4	3.260×10^{6}	3.639×10^{11}
kb2	43×84	61	33	2.478×10^{-2}	72.15	1.781×10^{8}	2.698×10^{16}
sc50b	50×98	49	14	7.226×10^{-1}	324.9	3.185×10^{7}	2.321×10^{19}
blend	74×157	92	36	1.497×10^{-2}	38.27	5.208×10^{8}	5.287×10^{31}
sc105	105×208	103	40	2.050×10^{-1}	708.9	1.395×10^{9}	8.908×10^{41}
scagr7	129×269	140	130	3.304	6552	1.236×10^{9}	8.945×10^{54}
sc205	205×408	236	114	5.187×10^{-2}	2380	8.891×10^{10}	1.867×10^{83}
beaconfd	173×435	102	65	3.716×10^{-2}	5607	2.798×10^{11}	1.172×10^{89}
lotfi	153×461	239	134	5.474×10^{-3}	4179	1.443×10^{12}	1.779×10^{91}
bore3d	233×548	200	63	2.091×10^{-4}	301.2	5.209×10^{12}	1.204×10^{113}
sctap1	300×780	328	277	1.154×10^{-8}	2.000	1.444×10^{15}	6.361×10^{159}
agg2	516×818	218	187	3.047×10^{-2}	1.676×10^{6}	8.060×10^{14}	3.933×10^{153}
scagr25	471×971	742	607	3.007×10^{-2}	1.118×10^{4}	4.964×10^{12}	5.212×10^{200}
standmps	467×1542	279	197	1.998×10^{-2}	101.1	7.718×10^{10}	7.525×10^{299}
seba	515×1543	648	465	2.950×10^{-2}	2268	1.542×10^{12}	2.144×10^{308}
gfrd-pnc	616×1708	1080	285	4.331×10^{-2}	2.919×10^{4}	2.030×10^{13}	1.387×10^{319}
ship04s	402×1860	328	280	1.535×10^{-2}	216.0	2.361×10^{11}	1.774×10^{318}
fit1p	627×2304	1994	1940	4.468×10^{-8}	33.07	4.144×10^{16}	1.027×10^{433}
shell	536×2311	710	461	1.000	2.843×10^{5}	9.573×10^{12}	1.023×10^{408}
scfxm3	990×2361	1407	1112	3.210×10^{-5}	9.590×10^{3}	2.661×10^{16}	4.968×10^{490}